

APPENDIX A

```
5  /* $Header: /cvs/RealJava/src/rjavar/java/com/softcom/realjava/plugins/RealTOC.java,v 1.17
1999/05/20 20:47:42 aw Exp $ */
// Copyright (c) 1998 SoftCom, Inc. All Rights Reserved.

10 package com.softcom.realjava.plugins;

15 import javax.swing.*;
import javax.swing.tree.*;
import java.awt.*;
import java.awt.event.*;
import java.io.IOException;
import java.net.URL;
import java.net.MalformedURLException;
import org.xml.sax.*;
import com.microstar.xml.SAXDriver;
20 import com.softcom.realjava.*;
import com.softcom.realjava.time.*;

25 /**
 * Synchronized table of contents plugin.
 * Displays a table of contents (TOC) whose nodes are highlighted
 * in sync with the presentation. Clicking on a node seeks the
 * presentation to that nodes time.
 * <P>
 * <CODE>RealTOC</CODE> understands a <CODE>URL</CODE> param which
30 * refers to an XML document specifying a hierarchical TOC.
 * <P>
 * Sample XML object element:
 * <P><TABLE BORDER=1><TR><TD>
 * <PRE>
35 * &lt;?xml version="1.0"?&gt;
 * &lt;object
 *   classid="com.softcom.realjava.plugins.RealTOC"
 *   archive="plugin.jar,sax.jar,aelfred.jar"
 *   width="250" height="235"
40 *   sync="1000"
 *   duration="900000"
 *   &gt;
 *   &lt;param name="URL" value="toc.xml"/&gt;
 * &lt;/object&gt;</PRE>
45 * </TD></TR></TABLE><P>
 * This is the <CODE>TOC</CODE> XML DTD.
```

```

* <CODE>BEGIN</CODE> and <CODE>END</CODE> are times specified
* in the format documented in <CODE>TimeSpanRegistry.parseTime()</CODE>.
* Both are optional, but if either is specified, then both must be specified.
* <P> <TABLE BORDER=1> <TR> <TD>
5   * <PRE>
* &lt;!ELEMENT TOC (NODE+)&gt;
* &lt;!ELEMENT NODE (TITLE, NODE*)&gt;
* &lt;!ATTLIST NODE
*      BEGIN CDATA #IMPLIED
10    END CDATA #IMPLIED
* &gt;
* &lt;!ELEMENT TITLE (#PCDATA)&gt; </PRE>
* </TD> </TR> </TABLE> <P>
* Sample <CODE>TOC</CODE> XML document:
15   * <P> <TABLE BORDER=1> <TR> <TD>
* <PRE>
* &lt;?xml version="1.0"?&gt;
* &lt;TOC&gt;
*   &lt;NODE BEGIN="0" END="4.999"&gt;
20     &lt;TITLE&gt;Root node one&lt;/TITLE&gt;
*   &lt;NODE BEGIN="5" END="9.999"&gt;
*     &lt;TITLE&gt;Node one child&lt;/TITLE&gt;
*     &lt;NODE BEGIN="10" END="14.999"&gt;
*       &lt;TITLE&gt;Node one subchild 1&lt;/TITLE&gt;
25   *     &lt;/NODE&gt;
*     &lt;NODE&gt;
*       &lt;TITLE&gt;Node one subchild 2&lt;/TITLE&gt;
*     &lt;/NODE&gt;
*   &lt;/NODE&gt;
*   &lt;/NODE&gt;
30   * &lt;/TOC&gt; </PRE>
* </TD> </TR> </TABLE> <P>
* RealTOC uses the AElfred XML parser and the SAX XML API.
* See <A TARGET="_top"
35 HREF="http://www.microstar.com/aelfred.html" > http://www.microstar.com/aelfred.html <
/A>
* for information on the AElfred XML parser.
* See <A TARGET="_top"
40 HREF="http://www.microstar.com/sax.html" > http://www.microstar.com/sax.html </A>
* for information on the SAX API to XML parsers.
*/
public class RealTOC extends JScrollPane implements Plugin {

    private static final String MESSAGE_CATALOG =
45        "com.softcom.realjava.plugins.RealTOCMessages";

```

```
private PluginContext m_pcContext;
private JTree m_jcTree;
private TimeSpanRegistry m_tsRegistry = new TimeSpanRegistry();

5    private static final String DTD = "toc.dtd";

    // Implements Plugin
10   public Synchronized getSynchronized() {
        return m_tsRegistry;
    }

    // Implements Plugin
15   public void startPlugin(PluginContext pc) {
        m_pcContext = pc;

20       // Construct AElfred SAX driver
        Parser parser = new SAXDriver();

        // Set parser to handle the XML document.
        parser.setDocumentHandler(new TOCParser());

        String strURL = m_pcContext.getParameter("URL");
        try {
            if (strURL == null)
                throw new MalformedURLException();
            URL url = new URL(m_pcContext.getDocumentBase(), strURL);
            InputSource is = new InputSource(url.openStream());

            // Pass URL to DTD
30            is.setSystemId(getClass().getResource(DTD).toString());

            // Parse the document
            parser.parse(is);
        } catch (MalformedURLException e) {
            Console.showConsole();
            System.err.println(MessageCatalog.getMessage(MESSAGE_CATALOG,
40            getClass(), "msg.inf.invalidURL", strURL));
        } catch (IOException e) {
            Console.showConsole();
            System.err.println(MessageCatalog.getMessage(MESSAGE_CATALOG,
45            getClass(), "msg.inf.parse"));
            e.printStackTrace();
        } catch (SAXException e) {
            Console.showConsole();
            System.err.println(MessageCatalog.getMessage(MESSAGE_CATALOG,
getClass(), "msg.inf.parse")));
        }
    }
}
```

```
    Exception ee = e.getException();
    if (ee == null)
        e.printStackTrace();
    else
        ee.printStackTrace();
5
    }

// Add the tree
10    getViewport().add(m_jcTree, BorderLayout.CENTER);
}

// Implements Plugin
public void destroyPlugin() {
}

15
// Object registered with event registry for each TOC event.
// This object is also set as the TreeNode userObject
private class TOCEvent implements TimeSpanListener {
    // Text to display in tree node
20    private String m_strText;

    // -1 if no seek time specified
    private int m_nTime = -1;

25    // Path to node this event is associated with
    private TreePath m_tpPath;

    public TOCEvent(DefaultMutableTreeNode tn) {
        m_tpPath = new TreePath(tn.getPath());
30    }

    // Set text to display in tree node
    void setText(String strText) {
        m_strText = strText;
    }

35    // Set time to seek media to tree node selected
    public void setTime(int nTime) {
        m_nTime = nTime;
    }

40    // Return seek time
    public int getTime() {
        return m_nTime;
    }

45}
```

```

// Event time reached, highlight tree node
// Implements TimeSpanListener
public void beginTimeSpan(TimeSpan ts) {
    // Select this tree node. This will make it visible too.
    m_jcTree.addSelectionPath(m_tpPath);
}

// Implements TimeSpanListener
10 public void endTimeSpan(TimeSpan ts) {
    // Deselect this tree node.
    m_jcTree.removeSelectionPath(m_tpPath);
}

// This is used by JTree to draw the node
15 public String toString() {
    return m_strText;
}
};

20 private class TOCParser extends HandlerBase {
    // Root of tree
    private DefaultMutableTreeNode m_tnRoot;
    // Current parent node
    private DefaultMutableTreeNode m_tnParent;
    // Current child node
    25 private DefaultMutableTreeNode m_tnCurrent;

    // Title text accumulator
    private StringBuffer m_sbText = new StringBuffer();
    private boolean m_bAccumulateText = false;

    // Overrides HandlerBase
    public void startElement(String strName, AttributeList attrs) throws SAXException
{
    30        if (strName.equals("TOC")) {
            // Create hidden root of tree
            m_tnRoot = m_tnCurrent = new DefaultMutableTreeNode();
        }
    35        else if (strName.equals("NODE")) {
            m_tnParent = m_tnCurrent;

            // Create a new node
            m_tnCurrent = new DefaultMutableTreeNode();
        }
    40        // Add node as a child of the current node
        m_tnParent.add(m_tnCurrent);
    45
}

```

```

// Create event for node
TOCEvent te = new TOCEvent(m_tnCurrent);
m_tnCurrent.setUserObject(te);

5

// Set time on event if specified
String strBegin = attrs.getValue("BEGIN");
if (strBegin != null) {
    String strEnd = attrs.getValue("END");
    if (strEnd == null)
        throw new
10
SAXException(MessageCatalog.getMessage(MESSAGE_CATALOG, getClass(),
"msg.ex.missingTime"));

15
try {
    int nBeginTime = TimeSpanRegistry.parseTime(strBegin);
    int nEndTime = TimeSpanRegistry.parseTime(strEnd);
    // Set seek time in event
    te.setTime(nBeginTime);
    // Register event with the TOC timespan registry
    m_tsRRegistry.addTimeSpan(new TimeSpan(te, nBeginTime,
20
nEndTime));
    } catch (NumberFormatException e) {
        throw new
25
SAXException(MessageCatalog.getMessage(MESSAGE_CATALOG, getClass(),
"msg.ex.invalidTime"), e);
    }
}
30
else if (strName.equals("TITLE")) {
    // Reset String Buffer for new title
    m_sbText.setLength(0);
    m_bAccumulateText = true;
}
35
// Invalid element
else
    throw new
SAXException(MessageCatalog.getMessage(MESSAGE_CATALOG, getClass(),
"msg.ex.invalidElement", strName));
40
}
// Overrides HandlerBase
public void endElement(String strName) throws SAXException {
    if (strName.equals("TOC")) {
        // Finished building tree. Setup JTree with model.
        // Create the tree with the constructed nodes
        m_jcTree = new JTree(m_tnRoot);
45
}

```

```

m_jcTree.setRootVisible(false);
m_jcTree.setShowsRootHandles(true);

// Allow multiple selection so we can support overlapping timespans
5

m_jcTree.getSelectionModel().setSelectionMode(TreeSelectionModel.DISCONTIGUOUS_TREE_SELECTION);

10 // When a tree node is clicked (regardless of whether it was selected),
// seek the video to the TOCEvent time registered with the node
m_jcTree.addMouseListener(new MouseAdapter() {
    // XXX mouseClicked is unreliable - it is not always called (bugid
4224704)

15 //public void mouseClicked(MouseEvent e) {
    public void mouseReleased(MouseEvent e) {
        // Only handle single clicks
        if (e.getClickCount() != 1)
            return;
        // Get the path and node clicked on
        TreePath path = m_jcTree.getPathForLocation(e.getX(),
20 e.getY());
        if (path != null) {
            // Get the event registered with this node
            TOCEvent te =
                (TOCEvent)((DefaultMutableTreeNode)path.getLastPathComponent()).getUserObject();
            if (te != null && te.getTime() >= 0) {
                // Seek the video
                m_pcContext.seekPlayer(te.getTime());
            }
30        }
    }
});

35 else if (strName.equals("NODE")) {
    // Back up a level
    m_tnCurrent = (DefaultMutableTreeNode)m_tnCurrent.getParent();
}
else if (strName.equals("TITLE")) {
    // Give title to current node
40 ((TOCEvent)m_tnCurrent.getUserObject()).setText(m_sbText.toString());
    m_bAccumulateText = false;
}
}

45 // Overrides HandlerBase
public void characters(char ch[], int nStart, int nLength) throws SAXException {

```

```
    if (m_bAccumulateText)
        m_sbText.append(ch, nStart, nLength);
    }
};

5
/*
// XXX debugging
public static void main(String args[]) {
    Frame frm = new Frame();
    frm.setLayout(new BorderLayout());
    RealTOC rt = new RealTOC();
    rt.startPlugin(new PluginContext() {
        public URL getCodeBase() {
            try {
15                return new
                URL("file:S:/projects/realjava/src/rjavar/plugclasses/");
            } catch (MalformedURLException e) {
                return null;
            }
        }
        public String getParameter(String strParam) {
            return "toc.xml";
        }
        public int getDuration() {
            return 0;
        }
        public void seekPlayer(int nTime) {
        }
        public void showDocument(URL url, String strTarget) {
30
    });
    frm.add(rt, BorderLayout.CENTER);
    frm.setSize(300, 300);
    frm.setVisible(true);
35
    }
}
*/
```